Preprint
UCRL-JC-144016

An Algorithm for Projecting Points onto a Patched CAD Model

W. D. Henshaw

This article was submitted to
10th International Meshing Roundtable
Newport Beach, CA
October 7-10, 2001

May 29, 2001

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

AN ALGORITHM FOR PROJECTING POINTS ONTO A PATCHED CAD MODEL

William D. Henshaw

Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551
henshaw@llnl.gov

ABSTRACT

We are interested in building structured overlapping grids for geometries defined by computer-aided-design (CAD) packages. Geometric information defining the boundary surfaces of a computation domain is often provided in the form of a collection of possibly hundreds of trimmed patches. The first step in building an overlapping volume grid on such a geometry is to build overlapping surface grids. A surface grid is typically built using hyperbolic grid generation; starting from a curve on the surface, a grid is grown by marching over the surface. A given hyperbolic grid will typically cover many of the underlying CAD surface patches. The fundamental operation needed for building surface grids is that of projecting a point in space onto the closest point on the CAD surface. We describe an fast algorithm for performing this projection, it will make use of a fairly coarse global triangulation of the CAD geometry. We describe how to build this global triangulation by first determining the connectivity of the CAD surface patches. This step is necessary since it often the case that the CAD description will contain no information specifying how a given patch connects to other neighbouring patches. Determining the connectivity is difficult since the surface patches may contain mistakes such as gaps or overlaps between neighbouring patches.

Keywords: CAD models, grid generation, overlapping grids, hyperbolic, surface grids

1. INTRODUCTION

In this paper we describe a fast method to generate hyperbolic surface grids on CAD geometries. We are motivated by the problem of grid generation on geometrical configurations defined by computer aided design (CAD) programs. In our approach we build a set of structured overlapping grids [6] that cover the computational domain. The grids are allowed to overlap which simplifies the grid construction process compared to the multi-block approach. The overlapping grids are connected through interpolation. The location of the interpolation points and "hole regions" (parts of grids that are unused) are computed automatically using the **Ogen** overlapping grid generator [7]. **Ogen** is part of the **Overture** object oriented framework which can be used to generate grids and solve partial differential equations [1, 2].

The first step in building overlapping volume grids is the generation of a set of overlapping surface grids. Both the surface and volume grids are typically generated using a hyperbolic marching algorithm [8]. The description of the geometry is often in the form of a collection of trimmed patches, see figure (1). The output from a CAD program will often be saved in a standard file format such as IGES or the newer STEP specification. The description defines a boundary-representation (B-REP) of the geometry, as opposed to say a solid-model representation. Unfortunately the typical IGES output file does not include any connectivity (topology) information, that is there is no information specifying how a given patch connects to other neighbouring patches. To further complicate matters the trimmed patches will often be inaccurate, or contain mistakes, making it difficult to determine where two neighbouring patches should be joined. As a first step in the grid generation process this connectivity information must be determined. As a second step we build a global triangulation of the surface where the triangulation will respect the boundaries of the trimmed patches. The fundamental operation needed for building surface grids is that of projecting a point in space onto the closest point on the CAD surface. The global triangulation will be used to aid in this projection step. In order to project a point onto the surface we first project onto the global triangulation and then project onto a particular surface patch.

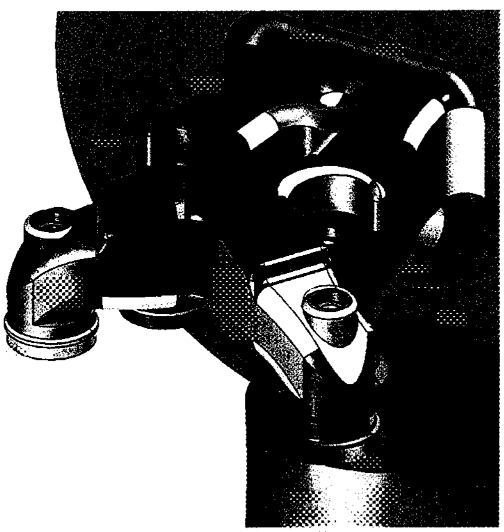


Figure 1. A CAD geometry represented as a collection of trimmed surface patches. No connectivity information is provided in the IGES file.

The approach we take to determine the connectivity of patched CAD model is based on the "Edge-Curve" approach described by Steinbrenner, Wyman and Chawner [11]. In this technique we first build curves (edge-curves) on the boundaries of all trimmed-patches and then attempt to identify where an edge-curve from one patch matches to the edge curve of a neighbouring patch. It is usually necessary to split the edge curves at appropriate locations in order to perform the matching. When two edge-curves are identified to be the same we say the edges have been merged and choose one edge-curve to define the boundary segment for both patches. By merging edge curves we effectively remove any gaps or overlaps present in the original representation. The details of algorithm described here differ in a variety of ways from that of Steinbrenner et.al. [11] such as in the representation of the edge curves, the order of the merging and splitting operations and the data structures used for searching. Once the edge curves have been matched we then can form a global triangulation for the patched surface. The first step in forming this global triangulation is to build a triangulation on each trimmed patch. The triangulation of each patch is performed in the two-dimensional parameter space, permitting the use of fast triangulation algorithms. The triangulation on each patch will have boundary nodes that are defined by the merged edge-curves. This means that the separate surface triangulations can be connected together since they will share boundary nodes with a neighbouring patch.

The global triangulation serves as a basis for a fast projection algorithm for projecting points onto the patched surface. This projection algorithm is used by the hyperbolic surface grid generator. The projection algorithm can also be useful for other purposes such building a high quality surface triangulation. To project a point onto the patched surface we first project the point onto the global triangulation. Finding the closest triangle is performed by a walking-algorithm if an initial guess is known or by a global search using an alternating-digital-tree (ADT) tree. Since each triangle belongs to just one sub-patch we can then project the point onto the sub-patch using Newton's method. The hyperbolic grid generator solves a set of hyperbolic equations to generate a surface grid starting from some initial curve. At each step, the positions of the new grid points are predicted from values of the current grid points and the normal to the surface. The predicted points are then projected onto the patch surface.

The algorithms we describe here have been implemented within the **Overture** object oriented framework and will be made available with the **Overture** software which can be obtained from `http://www.llnl.gov/casc/Overture`.

2. DETERMINING THE CONNECTIVITY OF A PATCHED SURFACE

A patched-surface consists of a set of sub-surfaces. There are often hundreds of sub-surfaces. A sub-surface may defined in a variety of ways such as

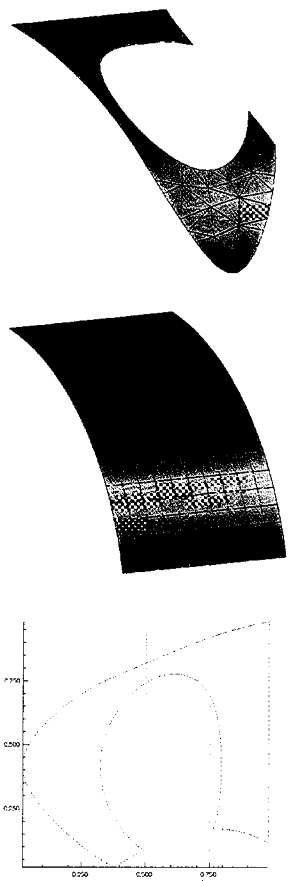


Figure 2. The trimmed patch (top) is formed from an untrimmed surface (middle) and a set of one or more trimming curves (bottom). The untrimmed surface is a mapping from two-dimensional parameter space into three-dimensional cartesian space. The trimming curves are defined in parameter space.

with a spline, B-spline or non-uniform-rational-bspline (NURBS). In general the sub-surface will be trimmed, in which case only a portion of the surface will be used, the valid region is defined by trimming curves, see figure (2).

It is often the case that the CAD file contains no topology information, that is there is no information to say which sub-surface connects to which other sub-surfaces. The purpose of the connectivity algorithm is to determine how the sub-surfaces are joined to one another. Once the connection information is computed a triangulation for the whole surface can be found.

A useful feature of the connectivity algorithm is that it will aid in the discovery of errors in the trimmed surfaces. Gross errors in the trimming curves are detected when the geometry is first read from the database file. Errors detected at this initial step include trim curves that lie outside the unit square in parameter space, trim curves that don't close on themselves (i.e. they should be periodic), and trim curves that self-intersect. These gross errors should be fixed before proceeding to the connectivity stage. In Overture we have the ability to edit the trim curves to fix these types of errors. Errors detected at the connectivity stage would include large gaps between patches or multiple definition of patches (sometimes the exact same trimmed patch may appear more than once in the CAD file!). These errors are usually easily found by visually inspecting the set of merged and unmerged curves. There should, for example, only be unmerged curves on the boundary of the surface.

There are two main steps in determining how sub-surfaces are connected:

build edge curves : build curve-segments that lie on the boundary of each sub-surface. A sub-surface defined by a NURBS, for example, will have 4 boundary curve segments. A sub-surface defined by a trimmed-mapping will have boundary segments corresponding to each trimming curve. A single trimming curve may be split into multiple boundary-segments, if the trimming curve was originally represented that way in the CAD file, or if the curve was split at corners.

merge/split edge curves : We examine the curve-segments to look for matching segments. If two segments agree (as a few number of points to some tolerance) we declare that the segments *are the same* (i.e. that they both represent the *true* boundary curve). Where two segments are the same, we also declare that their respective sub-surfaces are joined. It may be necessary to **split** a curve-segment into two or more pieces so that the pieces can be joined to other segments. After merging all possible curve-segments we should have matched all sub-surfaces where they join other sub-surfaces, thus determining the topology of the surface.

Trimming curves are usually defined in the two-dimensional parameter space of the patch. In order to compare edge-curves from different patches we must build three-dimensional representations for the edge curves. In some cases we can build an exact representation of the edge curve. For example, if the edge curve is a parameter line on a NURBS then the edge curve is itself a NURBS.

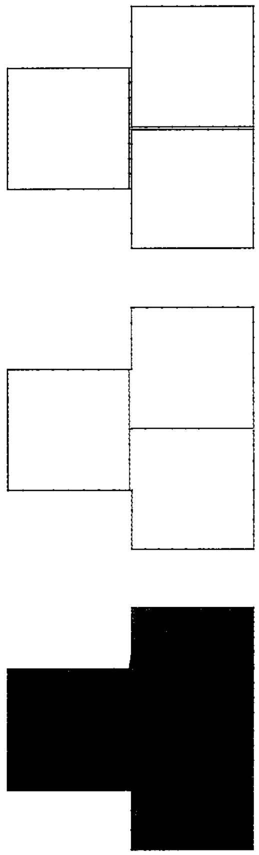


Figure 3. Figure showing the three stages of determining the connectivity. Edge curves are built on each side of each patch (top). The edge curves are merged and then split and merged (middle). Green curves have been merged, blue and red curves have not been merged. A red curve is an original curve that has been split. Triangulations are built separately for each patch and then stitched together at the common boundary points (bottom).

In other cases it would be too difficult to build an exact representation of the curve so instead we sample the curve at some appropriate number of points and then fit a curve to these points. The two-dimensional arclength and curvature of the curve are used to determine how many points to use. Usually we parameterize the 3D edge-curve using the parameterization of the 2D trimming curve unless the parameterization is poor and then we parameterize by the three-dimensional arclength. Usually a single trimming curve will be represented in the CAD file as a collection of sub-curves with each sub-curve being smooth. Normally when a trimming curve is created from a CAD file these sub-curves are merged into a single composite curve. In addition to the composite curve we also keep the the original sub-curves. These sub-curves will usually correspond to the curve of intersection between two surface patches and thus be exactly the edge-curves that we wish to merge. In some cases a trimming curve will not be smooth; a piece-wise linear NURBS can have sharp corners, and even higher-order NURBS can represent corners using multiple knots. Such trimming curves are split into smooth sub-curves by looking for multiple knots and detecting corners where the tangent changes rapidly.

After the edge curves have been built we then attempt to merge the edge curves. The merging step consists of two phases, see figure (3). In the first phase we examine all the original edge curves and look for matching curves. We use an alternating-digital-tree (ADT) to search for possible matching edge-curves. The ADT tree holds the bounding box for each edge curve. To determine if a given edge curve, $\mathbf{e}$, matches to some other edge curve we put a small box around one end-point of $\mathbf{e}$ and search for intersections with the bounding boxes of other edge curves. For any candidate edge-curve found in this way we then check more carefully that the curves agree at the end points and some number of interior points. In practice we have only found it necessary to compare the edge-curves at the end points and the midpoint. If two edges agree then we define the edges to be merged. One of the edge curves is defined to be the true edge curve. In the second phase we consider all curves that were not merged in the first phase. We attempt to split these curves into sub-curves which may then be merged. An edge curve can be split where it touches the end-point of another edge curve. For each un-merged edge we look for the endpoints of nearby edge-curves that will cause a split. The same ADT tree is used to locate edge curves whose end points could cause a split. A split is not allowed if the split position lies very close to the start or end of the un-merged edge.

We also tried another technique to determine the connectivity. In this alternative method we first triangulated each trimmed-patch and then attempted to stitch together neighbouring patches by adding boundary nodes of one patch-triangulation to a nearby patch-triangulation. This latter method worked reasonably well in many cases but ran into difficulties when the trimmed patches did not match very well (overlapping patches were especially troublesome) and for patches containing very thin regions. In contrast, the edge-curve merging approach works well since it assumes the boundaries of the trimmed-patches consist of piecewise smooth segments that should either match to a smooth segment of a neighbouring patch or be on the boundary of the surface. This assumption is correct for the CAD surfaces that we deal with.

3. BUILDING A GLOBAL TRIANGULATION

A global triangulation can be built once the edge curves have been merged. Recall that when two edge curves are merged, one of the two curves is defined to be the true edge curve. The global triangulation is formed by first triangulating each surface patch. The surface patches are triangulated in the parameter space of the patch. This allows us to use fast two-dimensional triangulation algorithms. We use the "triangle" program from Shewchuk [9] to compute a constrained Delaunay triangulation. It uses a divide and conquer algorithm to first build an unconstrained triangulation. The triangulation algorithm starts with a collection of edges that define the boundaries of the trimmed patch in parameter space. Each edge has two end-points. The end-points are taken from points on the trimming curve. The trimming curve will be defined in terms of the true edge curves computed in the merging step. This ensures that the boundary nodes of the triangulation of a patch will match to the boundary nodes of the triangulation of neighbouring patches. To improve the quality of the triangulation we initially add additional nodes to the interior of the triangulation and allow triangle to also add new nodes, however, we prevent new nodes from being added to the boundary. The quality of the triangles is also improved by scaling the parameter space coordinates, (r_0, r_1) by a transformation of the form $(\tilde{r_0}, \tilde{r_1}) = ((r_a + r_b r_1)r_0, (r_c + r_d r_0)r_1)$. The parameters r_a, r_b, r_c, r_d depend on the aspect ratio of the patch.

Since the merged edge-curve is defined in three-dimensional space we must determine the corresponding parameter space coordinates for nodes on the edge-curve. In some cases the parameter space coordinates are known from the time when the edge-curve was generated. In other cases we must project the 3D points onto the surface patch. In addition to being more expensive this projection step can also be error prone if the surface-patch is defined by a poor parameterization. For example, it it not uncommon that the surface has a coordinate singularity where one face is collapsed to a point. We double check the result of the projection step by comparing the projected 3D point to the original point being projected. If these points are not close we instead project the point onto the boundary edge of the surface.

Figure 4. Global triangulation for the diesel engine geometry. The triangulation respects the boundaries between the surface-patches and will be used to project points onto the original CAD representation.

After the patch has been triangulated in parameter space it is a simple matter to map the 2D parameter space nodes to 3D. The triangulations for the patches must be stitched together to form a global triangulation. We begin by joining the triangulations from the first two patches to form a valid global triangulation. The triangulation from patch three is then stitched to this global triangulation. The process is repeated until all triangulations have been added.

For each trimmed-patch we keep a list its trimming curves. Each trimming curve keeps pointers to the possibly two patches that use it (determined when the edge curves were merged). We use this information to determine whether a new patch is connected to any of the patches in the current global triangulation.

The triangulation of each patch is oriented so that the nodes of each triangle are ordered in a counter-clockwise order with respect to the parameter space triangulation. When a new patch triangulation is added it may be necessary to change the orientation of the triangulation to ensure that the normal to each triangle points in a consistent direction.

Figure (4) shows a global triangulation computed for a CAD description of a diesel engine. This example shows that a relatively coarse triangulation can be computed. The coarseness of the triangulation is determined by user-specified tolerance.

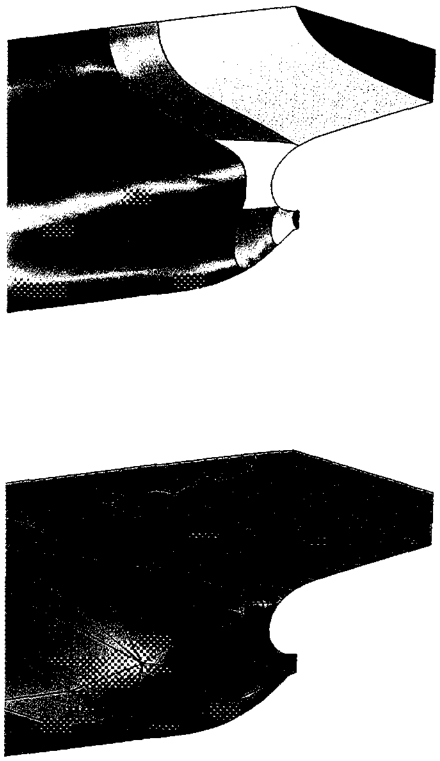


Figure 5. Surface patches and global triangulation for the stern of a tanker.

4. PROJECTING POINTS ONTO THE PATCHED SURFACE

The global triangulation for a patched surface can be used to define a fast projection algorithm. Given a target point, $\mathbf{p}_t$, in space near the surface we wish project the point onto the surface, i.e. we want to find the closest point on the surface to a given point, defined in some norm. This projection algorithm is used by the hyperbolic surface grid generator in Overture in order to build structured grids on the surface. The projection algorithm consist of two steps. First find the closest point on the triangulation. Secondly, uses the closest triangle to determine the closest surface patch and project onto the surface patch.

Given a good initial guess as to the closest triangle to $\mathbf{p}_t$, we find the closest point on the global triangulation we use a walking method. The walking method starts at a given triangle and marches to a neighbouring triangle that is closer to the target point. This marching continues until it reaches the boundary of the triangulation or else reaches an extremal triangle. An extremal triangle will be one where the line passing through the target point in the direction normal to the triangle face intersects the triangle. An extremal triangle could be a local maximum, a local minimum or a saddle point in the distance from the target point to the surface. We rely on the initial guess being good enough and the triangulation to be sufficiently fine for this walking method to give a reasonable answer.

If we do not have an initial guess we use a global search to find the closest point on the triangulation. The global search uses an alternating-digital-tree (ADT) tree in which we save the bounding boxes for all triangles on the global mesh. An ADT tree is a special type of binary search tree. We look for the intersection of a box around the target point with the triangle bounding boxes; this will determine potential triangles to check. The ADT tree is a fast way to answer this query. Given a list of potential triangles we check each one to determine the closest point. The only parameter in this search is the size of the bounding box around the target point. It should not be too large nor too small. We usually start with a safe value and then increase or decrease the box size depending on the number of intersections found.

Each triangle on the global triangulation lies on exactly one surface-patch. Once the closest triangle has been found we then suppose that the closest point on the surface will lie on the patch pointed to by the triangle. This assumes that the surface triangulation resolves the surface to a reasonable degree. If the target point lies very near the boundary between two patches it could be that the true projected point is on the neighbouring patch. For now we have ignored this possibility, although it would be possible to deal with this case. For our purposes so far it doesn't seem to be an issue.

5. HYPERBOLIC SURFACE GRID GENERATION

Structured surface grids can be generated using hyperbolic grid generation. This approach was developed by Steger, Chan and Buning [5, 3, 4] and is also available in Gridgen, see Steinbrenner and Chawner [10]. We have implemented our own version within the Overture framework [8]. Rather than have separate codes for surface and volume grid generation we instead have a single program that can generate 2D or 3D volume grids and 3D surface grids. The algorithms in all cases are basically the same. For surface grid generation we have the additional "boundary condition" that the grid points should lie on the defining boundary surface, which we denote by $\mathbf{C}(\mathbf{x}) = 0$.

Let (r, t) denote the parameter space (computational) coordinates. Instead of taking parameter space to be the unit cube we instead take the grid spacing in parameter space to be 1, $\Delta r = \Delta t = 1$. The basic marching equations to determine the surface grid $\mathbf{x}(r, t)$ given the initial curve $\mathbf{x}(r, 0)$ are defined by the hyperbolic PDE

$$\begin{aligned} \mathbf{x}_t &= S(r,t)\ \mathbf{n}(r,t) \\ \mathbf{x}(r,0) &= \mathbf{x}_0(r)\ ,\ \text{initial curve} \\ \mathbf{C}(\mathbf{x}(r,t) &= 0\ ,\ \text{grid is constrained to } \mathbf{C}(\mathbf{x}) = 0 \\ B(\mathbf{x}(r,t)) &= 0\ ,\ \text{boundary conditions} \end{aligned}$$

where

$$\begin{aligned} \mathbf{n}(r,t) &= \frac{\mathbf{x}_r \times \mathbf{n}_s}{\|\mathbf{x}_r \times \mathbf{n}_s\|}\ ,\ \text{normal to the front} \\ \mathbf{n}_s &: \ \text{normal to the surface } \mathbf{C} \text{ at } \mathbf{x} \\ S(r,t) &: \ \text{scalar speed function} \end{aligned}$$

and the norm $\|\cdot\|$ is defined by $\|\mathbf{f}\|^2 \equiv \mathbf{f}\cdot\mathbf{f}$. These equations march the grid in the direction locally orthogonal to the current front. The parameter t is a time like variable. At each step in time we generate a new grid line.

Note that the normal to the front, $\mathbf{n}$, is the marching direction for the front and should not be confused with the, $\mathbf{n}_s$, the normal to the surface we are marching over. The speed function $S(r, s, t)$ determines how fast the front propagates; it can depend on local properties of the front. Smoothing is also added to the equations so we actually solve a parabolic equation of the form

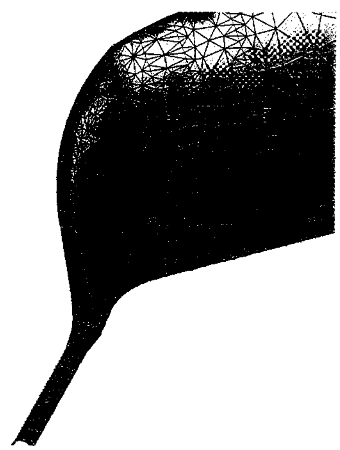
$$\mathbf{x}_t = S(r,t)\mathbf{n} + \epsilon(r,t)\mathbf{x}_{rr}$$

There are a variety of ways to define the speed function S. See [8] for some possible approaches.

These parabolic equations are solved in a fully implicit method with an approximate factorization requiring the formation and solution of a block tridiagonal matrix at each step.

Figure (6) show an example of growing a surface grid over a CAD surface. The user may optionally project onto the original CAD surface or simply project onto the triangulation. The original surface description could also just be a triangulation.

The projection algorithm is altered when the underlying surface is not smooth, to allow marching around corners.



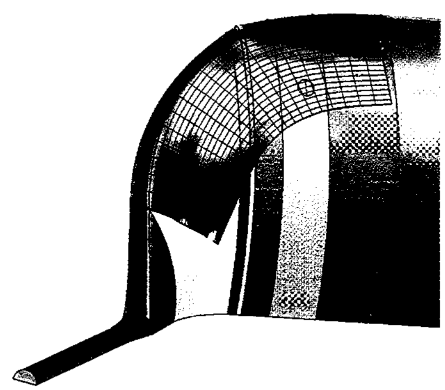


Figure 6. The hyperbolic surface grid generator uses the fast projection algorithm to grow surface grids. The triangulation is used to project the points onto the actual CAD geometry.

Acknowledgements: Thanks to Kyle Chand and Anders Petersson for help with the work presented here.

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

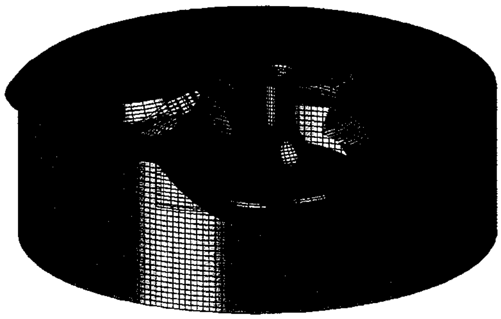


Figure 7. Overlapping grid generated on a CAD geometry. Most grids were generated with the hyperbolic grid generator.

REFERENCES

[1] D. L. Brown, W. D. Henshaw, and D. J. Quinlan, *Overture: An object-oriented framework for solving partial differential equations on overlapping grids*, in Object Oriented Methods for Interoperable Scientific and Engineering Computing, SIAM, 1999, pp. 245–255.

[2] ——, *Overture: Object-oriented tools for overset grid applications*, Research Report, for the AIAA conference on Applied Aerodynamics UCRL-JC-134018, Lawrence Livermore National Laboratory, 1999.

[3] W. Chan and P. Buning, *A hyperbolic surface grid generation scheme and its applications*, paper 94-2208, AIAA, 1994.

[4] W. M. Chan, *Hyperbolic methods for surface and field grid generation*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, 1999, ch. 5, pp. 1–26.

[5] W. M. Chan and J. L. Steger, *Enhancements of a three-dimensional hyperbolic grid generation scheme*, Applied Mathematics and Computation, 51 (1992), pp. 181–205.

[6] G. Chesshire and W. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys., 90 (1990), pp. 1–64.

[7] W. Henshaw, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[8] ——, *The overture hyperbolic grid generator, user guide, version 1.0*, Research Report UCRL-MA-134240, Lawrence Livermore National Laboratory, 1999.

[9] J. R. Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, in Applied Computational Geometry: Towards Geometric Engineering, M. C. Lin and D. Manocha, eds., vol. 1148 of Lecture Notes in Computer Science, Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[10] J. Steinbrenner and J. Chawner, *Gridgen's implementation of partial differential equation based structured grid generation methods*, in 8th International Meshing Rountable, 1998. www.andrew.cmu.edu/-user/sowen/imr8.html.

[11] J. Steinbrenner, N. Wyman, and J. Chawner, *Fast surface meshing on imperfect cad models*, in 10th International Meshing Rountable, 2000. www.andrew.cmu.edu/user/sowen/-imr9.html.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551